# SANDIA REPORT

SAND2015-9013
Unlimited Release
Printed October 2015

# Roadmap for Peridynamic Software Implementation

David J. Littlewood

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.



Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.



Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2015-9013
Unlimited Release
Printed October 2015

# Roadmap for Peridynamic Software Implementation

David J. Littlewood
Multiscale Science Department
Center for Computing Research
Sandia National Laboratories
P.O Box 5800
Albuquerque, NM 87185-1322

**Abstract**

The application of peridynamics for engineering analysis requires an efficient and robust software implementation. Key elements include processing of the discretization, the proximity search for identification of pairwise interactions, evaluation of the constitutive model, application of a bond-damage law, and contact modeling. Additional requirements may arise from the choice of time integration scheme, for example estimation of the maximum stable time step for explicit schemes, and construction of the tangent stiffness matrix for many implicit approaches. This report summaries progress to date on the software implementation of the peridynamic theory of solid mechanics. Discussion is focused on parallel implementation of the meshfree discretization scheme of Silling and Askari [33] in three dimensions, although much of the discussion applies to computational peridynamics in general.

# Acknowledgment

This report was supported by the Laboratory Directed Research and Development (LDRD) Program at Sandia National Laboratories. The author acknowledges many helpful discussions with Stewart Silling, John Mitchell, Michael Parks, John Foster, Dan Turner, and the *Sierra/SolidMechanics* development team.

# Contents


1 Introduction ..... 7
2 Evaluating the internal force density ..... 11
3 Bond damage and failure ..... 15
4 The tangent stiffness matrix ..... 17
5 Modeling contact ..... 21
6 Meshfree discretizations for peridynamics ..... 25
7 Proximity search for identification of pairwise interactions ..... 27
8 Time integration ..... 29
8.1 Explicit time integration for transient dynamics ..... 29
8.2 Estimating the maximum stable time step ..... 31
8.3 Implicit time integration for quasi-statics ..... 32
9 Example simulations ..... 35
9.1 Fragmentation of a brittle disk resulting from impact ..... 35
9.2 Quasi-static simulation of a tensile test ..... 36
10 Summary ..... 41

This page intentionally left blank.

# 1 Introduction

Peridynamics has advanced the state of the art for modeling material failure and fragmentation within a computational simulation code. To be effective in modeling real-world systems, the underlying theory must be accompanied by an effective software implementation. The goal of this report is to provide a roadmap for implementing peridynamics within an analysis code. This is motivated in part by the gaps that exist between much of the available literature on peridynamics, which is often focused on theoretical considerations, and implementation challenges that can arise in practice.

This report focuses exclusively on the meshfree approach of Silling and Askari [33]. To date, the vast majority of peridynamic simulations have utilized this approach. The decision to focus on this particular computational strategy is not meant to imply that it is superior to alternative approaches. Applying the finite-element technique to peridynamics, for example, may provide superior results in some cases. Nonetheless, the meshfree approach of Silling and Askari has proven to be a reliable and efficient strategy for addressing problems in solid mechanics with pervasive material failure and fracture. Advantages include ease of implementation, computational efficiency, and the presence of a natural mechanism for material separation. The method of Silling and Askari is a direct discretization of the strong form of the peridynamic balance of linear momentum [34],

$$\rho\left(\mathbf{x}\right)\ddot{\mathbf{u}}\left(\mathbf{x},t\right)=\int_{\mathcal{H}_{\mathbf{x}}}\left(\underline{\mathbf{T}}\left[\mathbf{x},t\right]\left\langle \mathbf{q}-\mathbf{x}\right\rangle -\underline{\mathbf{T}}\left[\mathbf{q},t\right]\left\langle \mathbf{x}-\mathbf{q}\right\rangle \right)dV_{\mathbf{q}}+\mathbf{b}\left(\mathbf{x},t\right), \tag{1}$$

where $\rho$ is the material density, $\ddot{\mathbf{u}}$ is acceleration, $t$ is time, $\mathbf{x}$ and $\mathbf{q}$ are material points, $\mathcal{H}_{\mathbf{x}}$ is a spherical neighborhood of radius $\delta$ centered at $\mathbf{x}$, $\underline{\mathbf{T}}$ denotes a peridynamic force state, $dV_{\mathbf{q}}$ denotes the infinitesimal volume associated with $\mathbf{q}$, and $\mathbf{b}$ is a body force density. For problems in three dimensional space, the approach of Silling and Askari requires evaluation of a three-dimensional integral. In contrast, solution of the weak form of the peridynamic balance of linear momentum, for example using a finite-element discretization, requires evaluation of a six-dimensional integral. For this reason, solution strategies for peridynamic models based on the weak form entail significant geometric complexity and computational expense relative to strong-form strategies.

The goal of a peridynamic simulation is to model material response within a body or a set of bodies under prescribed conditions over a specified period of time. This is accomplished numerically by discretizing the governing equations in both space and time. Following Silling and Askari [33], the physical system is discretized into a finite number of nodes, each having a (scalar) volume associated with it. A co-locational approach is employed, meaning that the nodes are used for tracking field variables such as displacement and velocity, and also as material points for constitutive model evaluations. This strategy is comparable to a one-point Gauss quadrature scheme in which all fields are assumed constant over the domain of integration and the Gauss point is located at the centroid of the domain (akin to midpoint integration).

The response of a peridynamic system as a whole is determined by solving the following

semidiscrete equation of motion over time for each material point in the discretization,

$$\rho\left(\mathbf{x}\right)\ddot{\mathbf{u}}\left(\mathbf{x},t\right)=\sum_{\mathcal{H}_{\mathbf{x}}}\left(\underline{\mathbf{T}}\left[\mathbf{x},t\right]\left\langle \mathbf{q}-\mathbf{x}\right\rangle -\underline{\mathbf{T}}\left[\mathbf{q},t\right]\left\langle \mathbf{x}-\mathbf{q}\right\rangle \right)\Delta V_{\mathbf{q}}+\mathbf{b}\left(\mathbf{x},t\right), \tag{2}$$

where the integral in Equation (1) has been replaced by a summation over the set of neighbors for material point $\mathbf{x}$. The force states $\underline{\mathbf{T}}\left[\mathbf{x},t\right]$ and $\underline{\mathbf{T}}\left[\mathbf{q},t\right]$ are determined through evaluation of the constitutive models at points $\mathbf{x}$ and $\mathbf{q}$, respectively. Contact interactions, if any, provide additional nodal force densities that must be included in Equation (2). In general, simulations may involve multiple bodies, multiple materials, and any number of specified initial and boundary conditions. Equation (2) holds for both bond-based and state-based constitutive models, as bond-based models can be considered a subset of the more general family of state-based models.

The progression of a simulation is driven by a time integration routine. The duration of the simulation is divided into a finite number of time steps or load steps at which Equation (2) is evaluated. In the case of the explicit time integration scheme described in Section 8.1, Equation (2) is applied to solve directly for nodal accelerations. The accelerations are then applied at each node to advance the simulation from one time step to the next. In contrast, the implicit time integration scheme for quasi-static simulations outlined in Section 8.3 treats the nodal displacements as the unknowns. Equation (2) is applied to solve for the displacement values that yield zero acceleration for those nodal degrees of freedom not subjected to kinematic boundary conditions. Regardless of the time integration strategy, simulations proceed by prescribing a set of know values at a given time step, *e.g.*, positions or velocities over a subset of the domain, followed by determination of the remaining field and material state data through application of the governing peridynamic equations. The net result is a complete description of kinematic and material state data for all material points (nodes) in the discretization for every time step in the simulation, from which any number of secondary quantities of interest may be determined.

At a high level, a computational peridynamics code contains several key components: a time integrator, a method for evaluating the internal force, routines for evaluating bond damage laws, routines for applying initial and boundary conditions, a means to evaluate and apply contact forces, and input/output routines for storing and retrieving data. For transient dynamic simulations, the critical time step must be estimated. Simulations employing implicit time integration generally require the implementation of a nonlinear solver. One standard approach for solving a nonlinear system of equations is Newton's method, which requires the construction of a tangent stiffness matrix and a method for solving a linear system of equations.

Peridynamics has been implemented in standalone software, and also as a component of larger, more comprehensive simulation codes. *EMU*, written by Stewart Silling, was the first computational peridynamics code [33]. Examples of preexisting software packages in which peridynamics has been implemented include *LAMMPS* and *Sierra/SolidMechanics* [23, 30]. It has also been demonstrated that peridynamics can be implemented in the commercial finite-element code *Abaqus* [20]. A more recent implementation of peridynamics is the open-source code *Peridigm* [24, 25].

There is potential for confusion with regard to the nomenclature applied to peridynamic discretizations. In this report, the terms *node*, *nodal volume*, and *material point* are used to describe a discretized peridynamic body. The term *nodal volume* refers directly to the pointwise position and volume data utilized in the discretization method of Silling and Askari. Nodes are used both as a means to track model geometry and as the locations for material model evaluations. The terms *node* and *material point* are used in reference to these distinct roles and provide alignment with standard computational mechanics terminology. The approach of Silling and Askari is typically referred to as *meshless* or *meshfree*, owning to the fact that nodes and nodal connectivities defining physical volumetric elements (as in standard finite-element approaches) are not considered. A primary source of confusion with regard to this terminology stems from the fact that, in practice, geometric data defining volumetric elements are often utilized in the discretization process as a means to compute nodal volumes (*cf.* Henke and Shanbhag [13]). Further, the geometries of the regions associated with each node have been utilized to improve the accuracy of peridynamic quadrature by taking into account the partial intersections of these entities with the spheres that define peridynamic neighborhoods [4, 29, 28].

The following sections address the primary components of a computational peridynamics code. Where it is illustrative to do so, the *Peridigm* code is used as an example. The discussion assumes that simulations are three dimensional, although concepts are generally applicable to one- and two-dimensional simulations as well. The majority of the content is focused on serial code. While implementation of a parallel code adds complexity, there are no fundamental changes required to the core peridynamic algorithms. Software implementation of a peridynamic constitutive model is presented first, followed by sections covering bond failure, the tangent stiffness matrix, contact modeling, meshfree discretizations, the proximity search for identifying peridynamic bonds, and time integration. Finally, example simulations demonstrating explicit transient dynamics and quasi-statics are presented.

This page intentionally left blank.

# 2 Evaluating the internal force density

The core kernels of a peridynamic code are the routines for computing the internal force density. The internal force density is determined via constitutive laws that compute pairwise force densities between bonded material points based on kinematic data and, optionally, material state variables. Software routines associated with a constitutive model include initialization, calculation of the internal force density, and, optionally, calculation of the tangent stiffness matrix. Each of these routines has access to initial positions, nodal volumes, neighbor lists, values for the current time and the time step size, and basic kinematic data that are updated throughout the simulation by the time integration routine (*e.g.*, velocities and current positions). In addition, constitutive models may define material-specific data fields that are tracked and updated over the course of a simulation.

Constitutive model routines are intimately tied to the time integrator. The goal of a time integrator is to advance a simulation from a given time step or load step, $n$, to the next step, $n+1$. For this reason, values corresponding to both step $n$ and step $n+1$ are tracked for many fields. Tracking nodal positions at steps $n$ and $n+1$, for example, enables a constitutive model to compute rates of deformation. The storing of solutions at both step $n$ and $n+1$ also facilitates the evaluation of trial configurations for implicit time integration. In this case, the solution at step $n$ must be available for reuse in the evaluation of multiple trial configurations.

The initialization and internal force density routines for the *linear peridynamic solid* constitutive model [34] with a Gaussian influence function are presented in Algorithms 1 and 2, respectively. Here, $\mathbf{x}$ denotes nodal coordinates in the undeformed configuration, $\boldsymbol{\xi}$ is a bond vector in the undeformed configuration, $\mathbf{u}$ denotes displacement, $\boldsymbol{\eta}$ denotes relative displacement of bonded material points, $\delta$ is the peridynamic horizon, $\underline{\omega}$ is an influence function value, $\underline{e}$ denotes extension state, $\underline{e}^d$ denotes deviatoric extension state, $m$ denotes weighted volume, $\Delta V$ is a nodal volume, $\theta$ denotes dilatation, $\underline{t}$ is a scalar force state, $\underline{\mathbf{M}}$ is the unit vector aligned with a bond in the deformed configuration, and $\mathbf{f}$ denotes internal force density. The material parameters $\mu$ and $k$ are the shear modulus and bulk modulus, respectively. Note that in the internal force density routine, the force state corresponding to each material point is computed only once. After the force state for a given point, $\mathbf{x}$, is computed, it is immediately applied to the bonds $\langle \mathbf{q}_i - \mathbf{x} \rangle$ and $\langle \mathbf{x} - \mathbf{q}_i \rangle$ that correspond to each neighbor, $\mathbf{q}_i$, of $\mathbf{x}$. The resulting pairwise force density vectors are summed directly into the global force density vector for both $\mathbf{x}$ and $\mathbf{q}$ in accordance with Equation (2). For simplicity, Algorithm 2 is presented with the assumption that the nodal volumes, $\Delta V_i$, are the full nodal volumes associated with node $i$. As discussed in Section 7, the physical volumetric region associated with a node may lie partially inside and partially outside a given peridynamic neighborhood. The accuracy of Algorithm 2 may be improved by modifying the nodal volumes to account for partial intersections (*cf.* Bobaru and Ha [4], Seleson [29], and Seleson and Littlewood [28]).

**Algorithm 1** The initialization routine for a *linear peridynamic solid* material with a Gaussian influence function.

**procedure** LINEAR PERIDYNAMIC SOLID INITIALIZATION
    ▷ Compute the weighted volume for each node.
    **for each** node $i$ **do**
        $m_i \leftarrow 0$
        **for each** node $j$ in neighbor list for node $i$ **do**
            $\boldsymbol{\xi} \leftarrow \mathbf{x}_j - \mathbf{x}_i$
            $\underline{\omega} \leftarrow \exp\left(-\frac{|\boldsymbol{\xi}|^2}{\delta^2}\right)$
            $m_i \leftarrow m_i + \underline{\omega}\,|\boldsymbol{\xi}|^2\,\Delta V_j$
        **end for**
    **end for**
**end procedure**

**Algorithm 2** Routine for calculation of the internal force density for a *linear peridynamic solid* material with a Gaussian influence function.

**procedure** LINEAR PERIDYNAMIC SOLID INTERNAL FORCE
    ▷ Initialize the global force density vector to zero.
    **for each** node $i$ **do**
        $\mathbf{f}_i \leftarrow 0$
    **end for**
    ▷ Compute the dilatation for each node.
    **for each** node $i$ **do**
        $\theta_i \leftarrow 0$
        **for each** node $j$ in neighbor list for node $i$ **do**
            $\boldsymbol{\xi} \leftarrow \mathbf{x}_j - \mathbf{x}_i$
            $\boldsymbol{\eta} \leftarrow \mathbf{u}_j - \mathbf{u}_i$
            $\underline{\omega} \leftarrow \exp\left(-\frac{|\boldsymbol{\xi}|^2}{\delta^2}\right)$
            $\underline{e} \leftarrow |\boldsymbol{\xi} + \boldsymbol{\eta}| - |\boldsymbol{\xi}|$
            $\theta_i \leftarrow \theta_i + \frac{3}{m_i}\, \underline{\omega}\, |\boldsymbol{\xi}|\, \underline{e}\, \Delta V_j$
        **end for**
    **end for**
    ▷ Compute the pairwise contributions to the global force density vector.
    **for each** node $i$ **do**
        **for each** node $j$ in neighbor list for node $i$ **do**
            $\boldsymbol{\xi} \leftarrow \mathbf{x}_j - \mathbf{x}_i$
            $\boldsymbol{\eta} \leftarrow \mathbf{u}_j - \mathbf{u}_i$
            $\underline{\omega} \leftarrow \exp\left(-\frac{|\boldsymbol{\xi}|^2}{\delta^2}\right)$
            $\underline{e} \leftarrow |\boldsymbol{\xi} + \boldsymbol{\eta}| - |\boldsymbol{\xi}|$
            $\underline{e}^d \leftarrow \underline{e} - \frac{\theta_i\, |\boldsymbol{\xi}|}{3}$
            $\underline{t} \leftarrow \frac{3}{m_i}\, k\, \theta_i\, \underline{\omega}\, |\boldsymbol{\xi}| + \frac{15\mu}{m_i} \underline{\omega}\, \underline{e}^d$
            $\underline{\mathbf{M}} \leftarrow \frac{\boldsymbol{\xi}+\boldsymbol{\eta}}{|\boldsymbol{\xi}+\boldsymbol{\eta}|}$
            $\mathbf{f}_i \leftarrow \mathbf{f}_i + \underline{t}\, \underline{\mathbf{M}}\, \Delta V_j$
            $\mathbf{f}_j \leftarrow \mathbf{f}_j - \underline{t}\, \underline{\mathbf{M}}\, \Delta V_i$
        **end for**
    **end for**
**end procedure**

This page intentionally left blank.

# 3 Bond damage and failure

Peridynamic bonds provide a natural framework for modeling material damage and failure. Following Silling and Askari [33], a scalar damage value is assigned to each bond in a peridynamic body. One approach is to define a damage variable, $\mathrm{d}_{ij}$, that tracks the damage between the bonded material points $\mathbf{x}_i$ and $\mathbf{x}_j$. The damage variable is restricted to values between 0.0 and 1.0, where $\mathrm{d}_{ij} = 0.0$ denotes an undamaged bond and $\mathrm{d}_{ij} = 1.0$ denotes complete bond failure. Bond damage values are updated throughout the course of a simulation and incorporated directly into the calculation of internal force density. For example, the influence function that defines a weighting between the bonded material points $\mathbf{x}_i$ and $\mathbf{x}_j$ in a state-based constitutive law may be modified by the factor $(1.0 - \mathrm{d}_{ij})$. In this way, the influence of damaged bonds is reduced, and bonds that have failed completely are effectively excluded from the simulation. The accumulation of broken bonds leads to material separation and the formation of cracks.

Pseudocode for the *critical stretch* bond failure law [33] is presented in Algorithm 3. The bond damage values, $\mathrm{d}_{ij}$, are initialized to 0.0 at the onset of the simulation. At each time step or load step, the bond stretch, $s$, defined as the change in bond length divided by the undeformed bond length, is computed for each pair of bonded material points in the body. The stretch values are then checked against a specified critical value, $s_o$. If the stretch exceeds the critical value, the bond is irreversibly broken ($\mathrm{d}_{ij}$ is set to 1.0).

As shown by Silling and Askari [33], the critical stretch value is directly related to the fracture energy associated with the creation of free surfaces,

$$s_o = \sqrt{\frac{5G_o}{9k\delta}}, \tag{3}$$

where $G_o$ is the fracture energy per unit area, $k$ is the bulk modulus, and $\delta$ is the peridynamic horizon. Additional connections between peridynamic bond failure laws and standard concepts in classical mechanics, including the formulation of a work-based bond failure criterion, are given by Foster, *et al.* [10].

**Algorithm 3** Routine for evaluation of the *critical stretch* bond failure law. Bond damage values, $\mathrm{d}_{ij}$, are initialized to zero at the beginning of the simulation and set to a value of one if the bond stretch exceeds the specified critical value.

**procedure** CRITICAL STRETCH BOND FAILURE
    **for each** node $i$ **do**
        ▷ Evaluate the stretch of each bond.
        **for each** node $j$ in neighbor list for node $i$ **do**
            $\boldsymbol{\xi} \leftarrow \mathbf{x}_j - \mathbf{x}_i$
            $\boldsymbol{\eta} \leftarrow \mathbf{u}_j - \mathbf{u}_i$
            $s = \frac{|\boldsymbol{\xi}+\boldsymbol{\eta}|-|\boldsymbol{\xi}|}{|\boldsymbol{\xi}|}$
            ▷ Check the bond stretch against the critical value.
            **if** $s \geq s_o$ **then**
                $\mathrm{d}_{ij} = 1.0$
            **end if**
        **end for**
    **end for**
**end procedure**

# 4 The tangent stiffness matrix

The tangent stiffness matrix is required when applying many nonlinear solver strategies for implicit time integration, such as Newton's method. It is comprised of the derivatives of nodal forces with respect to nodal displacements. Alternative formulations are also possible, for example one in which the nodal velocities are treated as the cardinal unknowns. A common approach for construction of the tangent stiffness matrix is to first consider the internal force (*i.e.*, the constitutive model response), after which the matrix is modified to reflect boundary conditions. For general nonlinear problems, the tangent stiffness matrix is a function of the chosen linearization point and must be re-evaluated whenever the nodal displacements are updated. It is important to note that several so-called linear peridynamic constitutive models impose a nonlinear relationship between force and displacement, and therefore require the application of a nonlinear solver. An example is the *linear peridynamic solid*, in which pairwise forces are linearly related to bond stretch but nonlinearly related to nodal displacements.

The tangent stiffness matrix is defined as

$$K_{ij} = \frac{\partial f_i^{int}(\mathbf{u})}{\partial u_j}, \tag{4}$$

where $f_i^{int}$ is a component of the internal force vector, $\mathbf{u}$ is the displacement vector, and $u_j$ is a component of $\mathbf{u}$. The tangent stiffness matrix relates an infinitesimal perturbation of a displacement degree of freedom to the resulting change in the global internal force vector. The tangent stiffness matrix for a nonlocal model contains nonzero entries for each pair of nodes that interact directly, and is inherently more dense than that of a local model. For a general state-based peridynamic model, direct interactions between nodes can take two forms: interactions between nodes that are bonded to each other, and interactions between nodes that share a common neighbor.

Methods for construction of the tangent stiffness matrix include both analytical and computational approaches. Analytical approaches may be developed using the concept of a peridynamic *modulus state*, as presented by Silling [32]. Examples of peridynamic constitutive models for which a modulus state has been derived include the *linear peridynamic solid* [34, 21], and the ordinary state-based viscoelasticity and plasticity models developed by Mitchell [21, 22]. When available, the modulus state provides a direct path to construction of the tangent stiffness matrix and is generally preferred over other approaches for reasons of accuracy and efficiency. Computational approaches for construction of the tangent stiffness matrix include finite-difference methods, described below, as well as methods that utilize software engineering techniques such as automatic differentiation [12] and the complex-step approach [6]. The *Peridigm* code includes a finite-difference scheme that is applicable to the full set of available constitutive models, as well as several material-model-specific routines that utilize the *Sacado* [27, 26] software package for automatic differentiation.

A finite-difference scheme for construction of the tangent stiffness matrix is presented in Algorithm 4. Finite-difference strategies are generic in that specifics of the constitutive

model formulation are not considered; all that is required is application of the constitutive model for evaluation of the internal force. Under the approach outlined in Algorithm 4, the partial derivatives in Equation (4) are approximated using a central-difference scheme,

$$K_{ij} \approx \frac{f_i^{int}\left(\mathbf{u}+\boldsymbol{\epsilon}^j\right) - f_i^{int}\left(\mathbf{u}-\boldsymbol{\epsilon}^j\right)}{2\epsilon}. \tag{5}$$

Here, $\boldsymbol{\epsilon}^j$ is a vector containing a single nonzero entry, $\epsilon$, at the position corresponding to the $j^{\text{th}}$ degree of freedom in the discretization. The notation used in Algorithm 4 to denote perturbed quantities is defined as follows: $\underline{\mathbf{T}}^{\epsilon+}$ is a force state evaluated under a positive perturbation of a displacement degree of freedom, $\underline{\mathbf{T}}^{\epsilon-}$ is the force state evaluated under the analogous negative perturbation, $\underline{\mathbf{T}}^{\epsilon+}\langle \mathbf{x}_k - \mathbf{x}_i \rangle$ and $\underline{\mathbf{T}}^{\epsilon-}\langle \mathbf{x}_k - \mathbf{x}_i \rangle$ are the force densities per unit volume resulting from the application of the perturbed force states to the bond $\langle \mathbf{x}_k - \mathbf{x}_i \rangle$, $\boldsymbol{f}^{\epsilon+}$ and $\boldsymbol{f}^{\epsilon-}$ are the corresponding nodal forces, and $\boldsymbol{f}^{\text{diff}}$ is the difference between $\boldsymbol{f}^{\epsilon+}$ and $\boldsymbol{f}^{\epsilon-}$. For an accurate approximation, the magnitude of $\epsilon$ should be chosen to be small relative to the discretization, but not so small that the limits of machine precision become a significant factor. The default perturbation size in *Peridigm* is $1.0e^{-6}$ times the nodal spacing.

Finite-difference schemes for constructing the tangent stiffness matrix involve perturbing individual displacement degrees of freedom and examining the resulting change in each component of the global force vector. To minimize computational expense, the evaluation of the internal force under a perturbation of a single displacement degree of freedom should be restricted to include only those components of the internal force vector that are directly affected. As described above, this subset of the internal force vector contains, at most, the union of the neighbor list of the perturbed node and the neighbor lists of each neighbor of the perturbed node.

The *Peridigm* algorithm for evaluating the tangent stiffness matrix by finite difference operates by traversing each node in the discretization and evaluating the force state at that node under a perturbation of a displacement degree of freedom. Perturbations must be considered for the node itself, as well as for each node in the node's neighbor list. Each time the force state is evaluated, it is applied to each bond in the neighbor list and multiplied by the volumes connected by that bond. The resulting force values are then divided by the perturbation length and assembled into the proper locations in the tangent stiffness matrix. An assumption implicit to this approach is that the peridynamic force state at a given material point can be determined based solely on the constitutive model data (and history) at that point, plus basic kinematic data (*e.g.*, position, velocity) at each neighbor. The approach utilized in *Peridigm* minimizes the number of internal force calculations required for construction of the tangent stiffness matrix by central finite difference. Alternative approaches, such as those based on graph coloring, provide additional strategies for evaluating Equation (5). It should be noted that the volumes $\Delta V_i$ and $\Delta V_j$ in Algorithm 4 must be treated in a manner consistent with evaluation of the constitutive model; if a correction for partial neighbor intersections is used in the constitutive model, it should applied for construction of the tangent stiffness matrix as well.

**Algorithm 4** Construction of the tangent stiffness matrix by central finite difference.

**procedure** TANGENT STIFFNESS MATRIX
  ▷ Initialize the tangent stiffness matrix to zero.
  $\mathbf{K} \leftarrow \mathbf{0}$
  ▷ Traverse each node in the discretization.
  **for each** node $i$ **do**
    $\{traversal\ list\} \leftarrow$ node $i$ and all neighbors of node $i$
    **for each** node $j$ in $\{traversal\ list\}$ **do**
      ▷ Evaluate the force state at $\mathbf{x}_i$ under perturbations of displacement.
      **for each** displacement degree of freedom $r$ at node $j$ **do**
        $\underline{\mathbf{T}}^{\epsilon+} \leftarrow \underline{\mathbf{T}}[\mathbf{x}_i](\mathbf{u} + \boldsymbol{\epsilon}^r)$
        $\underline{\mathbf{T}}^{\epsilon-} \leftarrow \underline{\mathbf{T}}[\mathbf{x}_i](\mathbf{u} - \boldsymbol{\epsilon}^r)$
        ▷ Evaluate pairwise forces under perturbations of displacement.
        **for each** node $k$ in neighbor list of node $i$ **do**
          $\boldsymbol{f}^{\epsilon+} \leftarrow \underline{\mathbf{T}}^{\epsilon+} \langle \mathbf{x}_k - \mathbf{x}_i \rangle \ \Delta V_i \ \Delta V_k$
          $\boldsymbol{f}^{\epsilon-} \leftarrow \underline{\mathbf{T}}^{\epsilon-} \langle \mathbf{x}_k - \mathbf{x}_i \rangle \ \Delta V_i \ \Delta V_k$
          $\boldsymbol{f}^{\text{diff}} \leftarrow \boldsymbol{f}^{\epsilon+} - \boldsymbol{f}^{\epsilon-}$
          **for each** degree of freedom $s$ at node $k$ **do**
            $\mathrm{K}_{sr} \leftarrow \mathrm{K}_{sr} + \frac{f_s^{\text{diff}}}{2\epsilon}$
          **end for**
        **end for**
      **end for**
    **end for**
  **end for**
**end procedure**

This page intentionally left blank.

# 5 Modeling contact

Modeling contact is important for many applications of peridynamics. Impact, penetration, and fragmentation simulations, for example, require the ability to capture multi-body contact interactions. This includes bodies that are disconnected at the onset of a simulation, and also those that become disconnected as a result of material failure. Contact modeling in peridynamics has received little attention relative to contact modeling for classical finite-element analysis, which has been addressed extensively in the literature (see, for example, Laursen [16]). To date, the most common approach to modeling contact in peridynamics is the *short-range force* approach of Silling and Askari [33]. The *short-range force* approach models contact interactions between bodies using a method similar to techniques employed in molecular dynamics. At each step in the simulation, spring-like repulsive forces are applied between nodes that are in close proximity to one another. This approach produces acceptable results in many cases. Alternatively, it has been demonstrated that a conventional (local) contact algorithm may be applied within a peridynamic simulation [17].

The basic components of a contact model are the detection algorithm and the enforcement algorithm. The detection algorithm performs a proximity search in the current (deformed) configuration. The proximity search required for contact is not different in principle from the proximity search required for the creation of neighbor lists. The *Peridigm* code executes the proximity search for contact in three phases: geometry-based parallel decomposition, identification and communication of off-processor data, and execution of an on-processor proximity search. The first two steps are required only for parallel simulations. Their goal is to reduce the parallel search to a set of serial searches. As described in Section 6, this is achieved by creating geometry-based parallel partitions and expanding the partitions in each direction by the contact search distance, thus collecting on each partition the necessary data to identify contact interactions for the on-processor nodes. A serial proximity search may then be carried out using a number of approaches, such as those based on quadtree or $k$-d tree data structures. Because nodes travel a finite distance over a time step, it is occasionally necessary to consider node paths as opposed to simply examining the node positions at the end of the step (*e.g.*, in hypervelocity simulations).

*Short-range force* algorithms consist of pairwise repulsive forces that increase in magnitude as the distance between nodes decreases. An extremely straightforward version of a *short-range force* calculation is illustrated in Algorithm 5. Here, $\mathbf{y}$ denotes nodal coordinates in the deformed configuration, $\ell$ is the distance between two nodes in the deformed configuration, $\ell_o$ is a threshold distance (*e.g.*, kick-in distance for short range forces), $C$ is a user-defined constant for controlling the magnitude of the short-range forces, $\mathrm{f}_c$ is a contact force magnitude per unit volume squared, $\underline{\mathbf{M}}$ is the vector between two nodes in the deformed configuration, and $\mathbf{f}^{\,\mathrm{contact}}$ denotes contact force density. Strengths of the *short-range force* approach include ease of implementation and relatively low computational cost. If desired, friction effects may be included through extensions to the *short-range force* model. The kinematic information required for a Coulomb friction model, for example, may be determined by examining the relative motion of any two nodes interacting via contact.

**Algorithm 5** A *short-range force* contact algorithm.

**procedure** SHORT-RANGE FORCE CONTACT ALGORITHM
  ▷ Initialize the contact force densities to zero.
  **for each** node $i$ **do**
    $\mathbf{f}_i^{\text{contact}} \leftarrow \mathbf{0}$
  **end for**
  ▷ Examine the results of the global proximity search for each node.
  **for each** node $i$ **do**
    $\mathbf{y}_i \leftarrow \mathbf{x}_i + \mathbf{u}_i$
    ▷ Compute pairwise contact force densities.
    **for each** node $j$ in proximity of node $i$ **do**
      $\mathbf{y}_j \leftarrow \mathbf{x}_j + \mathbf{u}_j$
      $\ell \leftarrow |\mathbf{y}_j - \mathbf{y}_i|$
      **if** $\ell < \ell_o$ **then**
        $\mathrm{f}_c \leftarrow \left(\frac{9\,C}{\pi\,\delta^4}\right)\,\left(\frac{\ell_o - \ell}{\delta}\right)$
        $\underline{\mathbf{M}} \leftarrow \frac{\mathbf{y}_j - \mathbf{y}_i}{\ell}$
        $\mathbf{f}_i^{\text{contact}} \leftarrow \mathbf{f}_i^{\text{contact}} - \mathrm{f}_c\,\Delta V_j\,\underline{\mathbf{M}}$
        $\mathbf{f}_j^{\text{contact}} \leftarrow \mathbf{f}_j^{\text{contact}} + \mathrm{f}_c\,\Delta V_i\,\underline{\mathbf{M}}$
      **end if**
    **end for**
  **end for**
**end procedure**

An additional extension of the *short-range force* model concerns peridynamic bonds. In its most simple form, the *short-range force* model does not take peridynamic bonds into account. This results in the possibility that material points may interact through the material model and the contact model simultaneously, for instance when a body is subjected to large hydrostatic compression. This effectively results in rapid material stiffening, often well beyond the stiffness prescribed by the constitutive model. This scenario can be avoided by considering neighbor list information in the contact model and disallowing contact forces between bonded nodes. For further discussion specific to correspondence constitutive models, in particular the possibility of material interpenetration, see Tupek and Radovitzky [36].

The computational expense of detecting and enforcing contact interactions often comprises a significant portion of the overall simulation cost. One strategy of reducing this expense is to limit the range of possible contact interactions to include only specific subregions or node sets. Further, so-called self contact may be disabled by designating that contact interactions may not occur between any two nodes that are within the same subregion.

A final consideration in contact modeling is the contact model's effect on the maximum stable time step for explicit time integration. The short-range force model, for example, typically results in a reduction of the critical time step due to what is effectively an increase in material stiffness in regions where contact is occurring. This can be problematic if the critical time step is sharply reduced when advancing the simulation from time $t^n$ to $t^{n+1}$ (see Section 8.1). Care must be taken to ensure that the time step applied at $t^n$ is sufficiently small to guarantee stability at $t^{n+1}$.

The *short-range force* algorithm does not perform well under all conditions. One source of difficulty is the tracking of surfaces. The *short-range force* model effectively represents surfaces as a collection of nodes, a strategy that breaks down as the distance between adjacent nodes grows large under finite deformation. This scenario is exacerbated in cases where the peridynamic discretization is derived from a nonuniform hexahedral or tetrahedral mesh (see Section 6). In this case, slender elements are replaced by nodes that yield poor representations of continuous surfaces. Poor surface approximations may lead to unphysical interpenetration and subsequent numerical difficulties. A possible solution is the construction of explicit surface representations, for example using a level-set approach (*cf.* Chi, *et al.* [7]).

Contact modeling in peridynamic simulations remains an open area of research. To date, contact has been applied primarily within explicit transient dynamic peridynamic simulations. A penalty- or constraint-based approach is more likely to be successful when implicit time integration is utilized. Furthermore, general issues related to nonlocality in contact have not been explored. For example, the interaction distance for the *short-range force* contact model ($\ell_o$ in Algorithm 5) is typically assigned a value approximately equal to the node spacing in the undeformed discretization, whereas the interaction distance for the constitutive model is equal to the horizon. In this scenario, the contact model is effectively a local model, in contrast to the constitutive law, which is nonlocal. The application of a local contact model within a peridynamic simulation is tantamount to applying local boundary conditions to a nonlocal model. A nonlocal contact model that produces force densities

acting over volumetric regions may be more appropriate for peridynamic simulations.

# 6 Meshfree discretizations for peridynamics

The meshfree approach of Silling and Askari requires that the domain of interest be discretized into a set of nodal volumes, each of which is defined by its initial coordinates, $(x, y, z)$, and volume, $\Delta V$. Nodal volumes may be grouped into a blocks, each tied to a specific material model and set of constitutive parameters. Standard *Peridigm* simulations operate on a discretization defined by tuples of the form $(x, y, z, \Delta V, block\_id)$. Discretizations of this type may be created internally within the code, read from a file, or created by processing a hexahedral or tetrahedral mesh (*e.g.*, a mesh created for use in a classical finite-element code). Several options for generating peridynamic discretizations are discussed in the recent work by Henke and Shanbhag [13], including the use of a Cartesian grid and a more sophisticated approach based on centroidal Voronoi tessellation. Each node in a discretization may be assigned a unique global ID. In the case of a parallel code, local (on-processor) IDs may also be assigned, and a mapping maintained between the local and global IDs. Unique global IDs facilitate the definition of node sets, to which initial and boundary conditions are applied. The use of node sets to facilitate the specification of initial and boundary conditions is commonplace in classical finite-element codes. A caveat regarding peridynamic models is that constraints are generally specified over a volumetric region, as opposed to a two-dimensional surface, which may complicate the specification of node sets in practice.

The use of blocks to define sets of nodal volumes facilitates simulations involving multiple materials and enables the application of specialized constitutive laws to bonds that cross material boundaries. Direct application of Equation (2) results in an averaging of constitutive responses when computing net pairwise interactions across material boundaries. Alternatively, a constitutive law that is specific to a material interface may be applied. Material damage along grain boundaries in a polycrystalline material, for instance, may be captured with a specialized bond failure law. In related work by Katiyar, *et al.* [15], the harmonic mean was shown to be an appropriate interface condition for modeling nonlocal diffusion.

Mesh generation tools developed for finite-element analyses can be applied to create discretizations for peridynamic simulations. Mesh generation software typically operates on a geometric description of the domain and produces a mesh of solid hexahedral or tetrahedral elements. Converting a mesh of this type into a meshfree discretization for a peridynamic simulation is straightforward: solid elements are converted to a $(x, y, z, \Delta V, block\_id)$ tuple, where $(x, y, z)$ is the centroid of the original element, $\Delta V$ is the volume of the original element, and the *block_id* is transferred directly from the original element to the peridynamic nodal volume. Calculation of the volume and centroid of a tetrahedron is trivial; for details regarding geometric calculations on hexahedral elements, see Grandy [11]. Conversion of a finite-element mesh in this fashion yields a one-to-one relationship between the elements in the original finite-element mesh and the nodal volumes in the meshfree peridynamic discretization. The nodes in the original mesh are not preserved, however, which complicates the transfer of node sets defined on the original mesh to nodes in the peridynamic discretization.

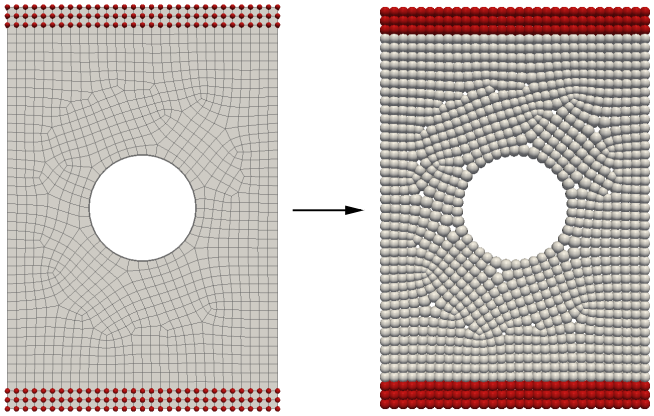


Figure 1: Conversion of a standard finite-element mesh to a meshfree discretization. Nodes belonging to a node set are shown in red. Nodal volumes in the meshfree discretization are visualized as spheres having volumes equal to the (scalar) nodal values.

It is important to recognize several differences between the requirements for a viable peridynamic discretization and those for a classical finite-element mesh. Boundary conditions in the classical local theory are applied over a two-dimensional surface, whereas nonlocal simulations require that constraints be applied over a three-dimensional volume. This complicates the definition of node sets for peridynamic simulations due to the fact that mesh-generation tools do not typically include functionality for defining three-dimensional layers along domain boundaries. The use of highly graded nodal spacing, which is common in classical finite-element analyses, is an additional source of difficulty. The use of a highly graded discretization in peridynamic simulations is problematic because the ratio of the nodal spacing to the horizon varies wildly over the domain. Computational expense can increase dramatically if the node spacing is reduced relative to the horizon size, which occurs in the highly-refined regions. A final point on the requirements for a viable peridynamic discretization concerns the chosen length scale. In many peridynamic constitutive models, the value of the horizon is raised to the fourth power, for example in the *prototype microelastic brittle* material [33]. For this reason, the unit of length should be chosen such that the value of the horizon is of order one to avoid potential numerical difficulties.

# 7 Proximity search for identification of pairwise interactions

A proximity search is required to identify the neighbors of the material points in a peridynamic body. A so-called neighbor list is a record of these points, for example a list of node IDs for all the nodes that comprise a given neighborhood. As a consequence of nonlocality, neighbor lists extend beyond the set of nearest neighbors to include all material points lying within a sphere centered at a given material point and having a radius equal to the peridynamic horizon. Note that, because the meshfree approach of Silling and Askari is a Lagrangian approach, neighbor lists do not change as a result of deformation and may be created and stored at the onset of a simulation [33].

The creation of neighbor lists, while straightforward in theory, presents several challenges in practice. For a general unstructured discretization, a global proximity search must be executed to find all material points within the sphere that defines the neighborhood of each material point in the model. For simplicity, the discussion below is restricted to the case in which a single horizon value is assigned to the entire computational domain. For a treatment of a variable horizon, see Bobaru and Ha [4] and Silling, *et. al* [35]. The *Peridigm* code utilizes the *Zoltan* software package for the creation of neighbor lists [5]. The *Zoltan* package contains routines with a variety of applications to parallel partitioning, load balancing, and data management, a number of which facilitate efficient proximity searches. Search algorithms based on quadtree or $k$-d tree data structures provide additional alternatives. In the case of a parallel code, proximity searches require an initial global decomposition phase. One strategy for this initial step is to create a geometrically-based parallel decomposition, for example using the Recursive Coordinate Bisection (RCB) algorithm [3]. When constructing neighbor lists by examining the distance between material points, expanding the bounding box for a given partition by a length equal to the horizon in each dimension allows for the identification of all potential off-processor neighbors to the material points in that partition.

Examining the distance between material points is not sufficient for quadrature schemes in which partial neighbor intersections are considered [4, 29, 28]. Following the approach of Seleson [29] and Seleson and Littlewood [28], for example, the domain is discretized into a set of subvolumes, each of which has a single peridynamic material point located at its centroid. The division of the domain into a set of subvolumes is analogous to the meshing procedure for classical finite-element analysis. In the case of finite-element analysis, the subvolumes are used directly as elements in the numerical simulation. In the case of a peridynamic simulation, the subvolumes are used only during pre-processing as a means to determine accurately the extent to which any pair of material points interacts. The pivotal case is the one in which the subvolume for a material point lies only partially within the sphere that defines the neighborhood for another material point. In this case, the intersection of the subvolume and the sphere may be computed and used in the calculation of pairwise internal forces (*e.g.*, through modification to $\Delta V_i$ and $\Delta V_j$ in Algorithm 2). Consideration of partial intersections adds significant complexity to the creation of neighbor lists. For general unstructured grids, the proximity search must take subvolume geometry into account. This

complicates both the serial proximity search and the geometric decomposition required for parallel codes. The serial proximity search must be capable of identifying and computing the intersections of subvolumes and spheres. The geometric decomposition required for parallel codes must employ an expanded bounding box that allows for identification of all potential off-processor partial intersections. Further, the results of the partial neighbor intersection calculations must be stored for use throughout the simulation, in addition to the standard $(x, y, z, \Delta V, block\_id)$ data.

Care must be taken when constructing neighbor lists in the vicinity of small geometric features. The danger is that bonds may pass across features that are small relative to the horizon size, for example small holes or notches, in a way that is unphysical. One possible solution is to include in the proximity search process a set of geometric entities through which bonds may not pass. For example, a finite plane may be used to define a pre-crack. Any bond that would pass through the pre-crack plane is excluded from the model. This approach can be made quite general in the case of a discretization created from a standard finite-element mesh. Shell elements, for instance, provide a means to define planes that can be used in the proximity search process [30]. An alternative, simpler approach to controlling the construction of neighbor lists is to explicitly allow or disallow bond creation between specific material subregions.

# 8 Time integration

Numerical time integration plays a central role in engineering analysis codes. Conceptually, it is the time integrator that drives the simulation and orchestrates the passing of information to and from various computational kernels. Time integration schemes for peridynamic models do not differ significantly from those of classical, local models, for which a vast literature exists. Standard approaches for engineering codes can be found in the excellent books by Hughes [14] and by Belytschko, *et al.* [2]. The time integration strategies for peridynamic models presented below were adopted directly from these texts.

Time integration schemes fall into two broad categories: explicit and implicit. Explicit time integration is typically utilized for transient simulations in which inertial effects play an important role. It is extremely resilient, and as such is well suited for peridynamic simulations involving large deformations and pervasive material failure. The great drawback of explicit time integration is that it is only conditionally stable, which limits the maximum allowable size of the time step. Implicit methods constitute a second class of time integration schemes. They are unconditionally stable, allowing for much larger time steps than those permitted by explicit approaches. Implicit time integration, however, requires the solution of a system of equations involving both the current state and a future state of the system. This adds considerable complexity and computational expense.

A discussion of explicit time integration for transient dynamics and implicit time integration for nonlinear quasi-static analysis is presented below. Far from a comprehensive treatment of numerical time integration, the intent is to highlight the most salient aspects of explicit and implicit methods for peridynamics. For details regarding other common time integration strategies, such as Newmark-Beta schemes for implicit dynamics, see Belytschko, *et al.* [2].

Note that, in some cases, it may be advantageous to apply multiple time integrators, in sequence, within a single simulation. An example is mechanical or thermal preloading, which may be simulated as a quasi-static process, followed by the simulation of a dynamic process such as impact.

## 8.1 Explicit time integration for transient dynamics

Generally, explicit time integration schemes determine system response over the course of a simulation by advancing through a large number of small time steps. At each step, known quantities in the current state of the system are used to determine the future state of the system. An explicit time integration scheme well suited for application to peridynamics is outlined below. It is a central difference scheme in which velocities are defined at the midpoints of the time intervals [2].

1. Initialize $n = 0$, $t = 0$, $\mathbf{u} = \mathbf{0}$, and $\mathbf{a} = \mathbf{0}$. Set initial conditions for $\mathbf{v}$ and initialize material state variables (if any).

2. Estimate the maximum stable time step, $\Delta t_{\text{crit}}$, and determine the simulation time step, $\Delta t$.

3. Update the times $t^{n+1}$ and $t^{n+\frac{1}{2}}$: $\quad t^{n+1} = t^n + \Delta t, \quad t^{n+\frac{1}{2}} = \frac{1}{2}\left(t^n + t^{n+1}\right)$.

4. First partial velocity update, $\mathbf{v}^{n+\frac{1}{2}} = \mathbf{v}^n + \left(t^{n+\frac{1}{2}} - t^n\right)\mathbf{a}^n$.

5. Enforce velocity boundary conditions by setting $\mathbf{v}^{n+\frac{1}{2}}$ for nodes with prescribed velocities. Displacement boundary conditions can be converted to velocities and applied here, or they can be applied directly after step 7.

6. Update nodal displacements, $\mathbf{u}^{n+1} = \mathbf{u}^n + \mathbf{v}^{n+\frac{1}{2}}\Delta t$.

7. Evaluate the internal force by means of the constitutive model and damage model for $\mathbf{u}^{n+1}$ and $\mathbf{v}^{n+\frac{1}{2}}$. Update material state variables accordingly. Optionally, recompute the estimate for the critical time step, $\Delta t_{\text{crit}}$, and update the simulation time step, $\Delta t$.

8. Evaluate contact forces for $\mathbf{u}^{n+1}$ and $\mathbf{v}^{n+\frac{1}{2}}$. Update the critical time step, $\Delta t_{\text{crit}}$, if required by the contact algorithm.

9. Compute acceleration $\mathbf{a}^{n+1} = \mathbf{M}^{-1}\boldsymbol{f}^{\,n+1}$, where $\mathbf{M}$ is the mass matrix and $\boldsymbol{f}^{\,n+1} = \boldsymbol{f}_{int}^{\,n+1} + \boldsymbol{f}_{ext}^{\,n+1} + \boldsymbol{f}_{contact}^{\,n+1}$.

10. Second partial velocity update, $\mathbf{v}^{n+1} = \mathbf{v}^{n+\frac{1}{2}} + \left(t^{n+1} - t^{n+\frac{1}{2}}\right)\mathbf{a}^{n+1}$.

11. Evaluate energy balance as a check on stability.

12. If the simulation is not complete, update $n \leftarrow n+1$, go to Step 3.

The first step in the time integration scheme is initialization. The time, $t$, and time step, $n$, as well as nodal values for displacement, $\mathbf{u}$, and acceleration, $\mathbf{a}$, are set to zero. The nodal velocities, $\mathbf{v}$, are be set to reflect initial conditions on the velocity field. Variables for representing material states within a constitutive model must also be initialized in Step 1. The maximum stable time step, also referred to as the critical time step, $\Delta t_{\text{crit}}$, is determined in Step 2, as described below. The time step, $\Delta t$, is typically determined by multiplying the critical time step by a safety factor. To guard against numerical instability, the safety factor must be less than 1.0; typical values fall in the range of 0.70 to 0.95. A constant value of $\Delta t$ may be applied throughout the course of the simulation, or, alternatively, the time step may be recomputed as part of Steps 7 or 8 to reflect evolving model geometry, material state, and contact interactions.

The time-stepping process begins with Step 3, in which both the time at end of the step, $t^{n+1}$, and the mid-step time, $t^{n+\frac{1}{2}}$, are computed. The midstep velocity is then determined in Step 4 using the current values for velocity and acceleration. Step 5 concerns the application of prescribed displacements and velocities, which are typically assigned to node sets that define volumetric subregions within the computational domain. Prescribed velocities may

be applied directly. Prescribed displacements may be converted to velocities through the relation

$$\mathbf{v}^{n+\frac{1}{2}} = \frac{1}{\Delta t}\left(\mathbf{u}^{n+1} - \mathbf{u}^{n}\right). \tag{6}$$

The mid-step velocities, $\mathbf{v}^{n+\frac{1}{2}}$, are then applied to determine the displacements at the end of the time step, $\mathbf{u}^{n+1}$, in Step 6.

The constitutive model, damage model, and contact model are invoked in Steps 7 and 8 for evaluation of the nodal forces, $\boldsymbol{f}^{n+1}$. The inputs to these models are the displacements at the end of the time step, $\mathbf{u}^{n+1}$, and the mid-step velocities, $\mathbf{v}^{n+\frac{1}{2}}$, as well as the material state variables stored at step $n$. The damage law may be applied first, to determine the current state of damage for each bond, followed by evaluation of the constitutive model. Contact algorithms, if any, are generally not tied directly to the constitutive model and may be evaluated independently. Contact forces, constitutive model forces, and prescribed body forces are summed together to determine the net force on each node.

Following determination of nodal forces, the nodal accelerations, $\mathbf{a}^{n+1}$, are computed by dividing by the mass associated with each node. The resulting acceleration values are then applied to determine the velocities at the end of the time step, $\mathbf{v}^{n+1}$, in Step 10. The time stepping process then returns to Step 3 and repeats until the simulation is complete.

Prior to incrementing to the next time step, it may be advantageous to carry out a check on global energies to ensure that the simulation remains numerically stable (Step 12). A check on the global energy balance typically involves integration of the kinetic energy, external work, and stored elastic energy over the computation domain (*cf.* Equation (62) in Silling, *et al.* [34]). An unphysical increase in the total energy of the system generally indicates an unstable simulation, for example resulting from an inaccurate estimate of the critical time step. Note that simulations involving crack formation necessarily absorb energy through the creation of new surfaces. Care must be taken in this case because numerical instability may manifest as the creation of additional (unphysical) cracks, as opposed to an increase in kinetic energy or stored elastic energy.

## 8.2 Estimating the maximum stable time step

A means for estimating the critical time step for the *prototype microelastic brittle* model was published by Silling and Askari [33]. At each material point in the discretization, a value for the maximum stable time step is determined as follows,

$$\Delta t_{\text{crit}} = \sqrt{\frac{2\rho}{\sum_p \Delta V_p C_p}}, \tag{7}$$

where $\rho$ is the density, $p$ iterates over all the neighbors of the given material point, $\Delta V_p$ is the volume associated with neighbor $p$, and $C_p$ is the micromodulus between the given material point and neighbor $p$. The minimum value over all material points in the discretization is

taken as the critical time step for the simulation. Equation (7) was derived for the one-dimensional case. Its application in two- and three-dimensional simulations results in a conservative estimate of the time step. (Effectively, it assumes that all bonds for a given material point are colinear.)

It has been demonstrated that Equation (7) may provide a reasonable estimate of the maximum stable time step for other constitutive models, for example the state-based *linear peridynamic solid* [19]. In this case, the micromodulus, $C_p$, appearing in the denominator of Equation (7) is replaced by an effective micromodulus,

$$C_p^{\text{eff}} = \frac{1}{|\boldsymbol{\xi}|} \frac{18\,k}{\pi\delta^4}, \tag{8}$$

where $|\boldsymbol{\xi}|$ is the distance between the given material point and neighbor $p$ in the reference configuration, $k$ is the bulk modulus, and $\delta$ is the horizon. The second term in Equation (8) is taken from Silling and Askari [33]; it is the spring constant for a *prototype microelastic brittle* material having bulk modulus $k$ and horizon $\delta$.

Another option for estimating the maximum critical time step is the well-known Courant-Friedrichs-Lewy (CFL) approach [8],

$$\Delta t_{\text{crit}} = \frac{h}{c}, \tag{9}$$

where $h$ is a characteristic length associated with the discretization and $c$ is the wave speed,

$$c = \sqrt{\frac{k}{\rho}}. \tag{10}$$

Equation (9) generally yields a very conservative estimate of the critical time step for peridynamic models when $h$ is set equal to the node spacing. This is because wave propagation in peridynamic models is largely dictated by the size of the horizon. The CFL limit is commonly applied in classical finite-element analyses, where nodes interact directly only when connected by a common element. In contrast, nodes in a peridynamic simulation may interact directly when separated by as much as twice the horizon. A natural question regarding Equation (9) is, can the horizon be used as the value for $h$? In preliminary numerical experiments, this approach yielded a nonconservative critical time step [19].

## 8.3 Implicit time integration for quasi-statics

Quasi-static analysis is appropriate for simulations in which relatively large load steps are desirable and in which dynamic effects are negligible. The governing equation for quasi-static simulations is obtained by setting the acceleration term to zero in Equation (2),

$$\sum_{\mathcal{H}_{\mathbf{x}}} \left( \underline{\mathbf{T}}\left[\mathbf{x}, t\right] \langle \mathbf{q} - \mathbf{x} \rangle - \underline{\mathbf{T}}\left[\mathbf{q}, t\right] \langle \mathbf{x} - \mathbf{q} \rangle \right) \Delta V_{\mathbf{q}} + \mathbf{b}\left(\mathbf{x}, t\right) = \mathbf{0}. \tag{11}$$

A quasi-static simulation consists of a series of load steps at which static equilibrium is enforced. Dynamic (inertial) phenomena, such as wave propagation, are not considered. At the onset of each load step, the boundary conditions are updated. In contrast to explicit transient dynamics, this update constitutes an increase in the applied load, as opposed to an increment of a physically-meaningful time parameter. Updating the boundary conditions generally results in an unequilibrated system configuration. The challenge is then to determine the configuration of the degrees of freedom not subjected to kinematic boundary conditions such that equilibrium is re-established.

The equilibrium configuration for a given load step is determined by minimizing the nodal forces. This process focuses on the residual vector, $\mathbf{r}$, which is defined as the nodal force vector with the degrees of freedom subject to kinematic boundary conditions removed or set to zero. In a perfectly equilibrated configuration, the residual vector is equal to $\mathbf{0}$. A scalar-valued residual, $r$, is defined as a means to track the incremental progress of the nonlinear solver as $\mathbf{r}$ approaches $\mathbf{0}$. The scalar-valued residual may be defined, for example, as the $\ell_2$-norm or $\infty$-norm of $\mathbf{r}$. A load step is complete when $r$ has dropped below a specified threshold value. The chosen form of $r$ and the specified threshold value define the convergence criterion for the nonlinear solver. For additional discussion of convergence criteria for the nonlinear solution process, see Belytschko, *et al.* [2].

The solution procedure outlined below for nonlinear quasi-static simulations is valid for both linear and nonlinear problems. For linear problems, however, the iterative procedure for reduction of the residual is not required and may be replaced with a single solve of the global linear system, given in Step 5b, below. In the linear case, a single Newton step, $\Delta\mathbf{u}$, is guaranteed to re-establish static equilibrium.

The solution process for general, nonlinear quasi-static peridynamic simulations using Newton's method as the nonlinear solver may be written as follows:

1. Initialize $n = 0$, $\mathbf{u} = \mathbf{0}$, and $\mathbf{v} = \mathbf{0}$. Initialize material state variables.
2. Update the load step $n \leftarrow n + 1$ and pseudo-time $t$. Update the boundary conditions.
3. Evaluate the residual vector, $\mathbf{r}$, and residual $r$. Determine the convergence criterion for the load step.
4. Assign an initial guess to the trial displacement $\mathbf{u}_{\text{trial}}$ (for example, $\mathbf{u}_{\text{trial}} = \mathbf{u}^n$).
5. Apply Newton's method to minimize the residual.
   (a) Construct the tangent stiffness matrix, $\mathbf{K}$, for the configuration $\mathbf{u}_{\text{trial}}$ (recompute, reuse, or modify).
   (b) Solve the linear system $\mathbf{K}\,\Delta\mathbf{u} = -\mathbf{r}$ for the Newton step, $\Delta\mathbf{u}$.
   (c) Set $\mathbf{u}_{\text{trial}} = \mathbf{u}_{\text{trial}} + \Delta\mathbf{u}$.
   (d) Evaluate the residual vector, $\mathbf{r}$, and the residual, $r$, for the updated configuration $\mathbf{u}_{\text{trial}}$.

(e) If the convergence criterion is not met, return to 5a.

6. Set $\mathbf{u}^{n+1} = \mathbf{u}_{\text{trial}}$. Set material state data to their trial values.

7. If simulation not complete, go to 2.

The simulation begins with field initialization, as well as the initialization of material state variables associated with the constitutive models. It is assumed that the system is in equilibrium at the onset of the simulation. Load stepping begins with Step 2, in which an increment of the boundary conditions is applied. This may take the form of updated displacement values in the case of kinematic boundary conditions, or updated external force values in the case of prescribed body forces. The extent to which the system is no longer in equilibrium is determined through the residual evaluation in Step 3. As discussed above, the residual is a function of the forces corresponding to degrees of freedom not subjected to kinematic boundary conditions. (The forces corresponding to degrees of freedom at which kinematic boundary conditions are applied are generally nonzero and are referred to as reaction forces.) The convergence criterion for a given load step is typically computed relative to the initial residual value, although an absolute criterion may also be used.

Solution of the nonlinear problem begins with Step 4. The goal is to determine a displacement vector increment, $\Delta\mathbf{u}$, that will reduce the residual such that the convergence criterion is met. While many techniques exist for solving nonlinear problems, Newton's method is perhaps the most straightforward. Here, the residual is minimized by solving a series of linear problems. The linear problems are defined as

$$\mathbf{K}\,\Delta\mathbf{u} = -\mathbf{r}, \tag{12}$$

where $\mathbf{K}$ is the tangent stiffness matrix, $\mathbf{r}$ is the residual vector, and $\Delta\mathbf{u}$ is an (unknown) increment in the nodal displacements. The tangent matrix is a linearization of the system about the current trial displacement, $\mathbf{u}_{\text{trial}}$, as described in Section 4. Prior to solving the linear system in Equation (12), the tangent stiffness matrix and right-hand side vector are typically modified to reflect kinematic boundary conditions. One strategy is to set the rows and columns of the stiffness matrix corresponding to kinematic boundary conditions to zero, with a scalar on the diagonal (*e.g.*, the $\ell_2$-norm of the unmodified diagonal). The entries in the residual vector corresponding to kinematic boundary conditions are also set to zero. The result of these modifications is that the solution of Equation (12) yields a zero increment in displacement for degrees of freedom corresponding to kinematic boundary conditions.

A quasi-static load step is deemed to be converged with the residual drops below a specified threshold value. The solution procedure then advances to Step 6, in which both the field variables and the material state variables are set equal to their trial values. The load stepping process is then repeated until the simulation is complete.

# 9 Example simulations

Results of an impact and fragmentation simulation and the results of a tensile bar simulation are presented in the following sections. Taken together, they are realizations of the numerical and computational strategies discussed above for evaluation of peridynamic constitutive models, calculation of the tangent stiffness matrix, contact modeling, and explicit and implicit time integration. The first is an explicit transient dynamics simulation of fracture that results from a projectile impacting a brittle disk. It is an initial value problem in which an initial velocity is assigned to the projectile. The second is a quasi-static simulation of a bar loaded in tension. In this case, the simulation is driven by prescribed displacement boundary conditions applied to the end portions of the tensile specimen. Both simulations are fully three-dimensional and were carried out using *Peridigm* [24, 25].

## 9.1 Fragmentation of a brittle disk resulting from impact

A peridynamic simulation of a fragmenting brittle disk is illustrated in Figure 2. It is a modified version of the brittle fragmentation simulation presented by Silling and Askari [33], variations of which were later carried out by Parks, *et al.* [23], and Henke and Shanbhag [13]. The simulation is presented for the purpose of demonstrating explicit time integration for transient dynamics (Section 8.1), contact (Section 5), and crack formation via bond failure (Section 3). For a more complete treatment of modeling brittle fragmentation with peridynamics, including an investigation of convergence with respect to nodal spacing, horizon size, and time step size, see Henke and Shanbhag [13].

The disk and projectile were modeled using the *linear peridynamic solid* constitutive model [34], and material failure in the disk was modeled with the *critical stretch* bond failure law [33]. Constitutive model parameters are given in Tables 1 and 2. The geometry of the disk is a cylinder of radius 37.0 mm and height 2.5 mm. The discretization for the disk was created by converting an unstructured hexahedral mesh, created with the *Cubit* mesh generation code [9], into a meshfree discretization containing approximately 180 000 nodes. The geometry of the projectile is a sphere with radius 5.0 mm. The discretization for the projectile was created by converting a tetrahedral mesh, created with *Cubit*, into a meshfree representation containing approximately 18 000 nodes. The node spacing for both the disk and the projectile was set to roughly one third of the peridynamic horizon. Contact was modeled with a *short-range force* contact model, as described in Section 5. The contact radius was set to 0.775 mm and the spring constant to 1000.0 GPa. The disk was initially at rest, and the projectile was assigned an initial velocity of 100.0 m/s. The simulation was run to a final simulation time of 400.0 µs. A constant time step of 0.0416 µs (0.7 times the estimated critical time step) was used throughout the simulation. The fragmenting disk simulation was run in parallel using ten 3.0 GHz Intel Xeon processors, requiring a total run time of approximately 2.8 hours.

Table 1: Constitutive parameters for the disk, modeled using the *linear peridynamic solid* constitutive model and the *critical stretch* bond failure law.

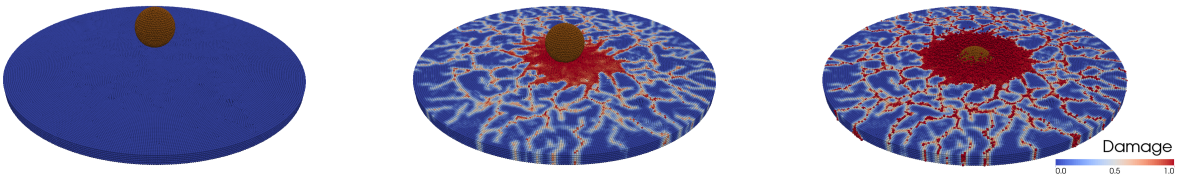
| Parameter | Value |
|---|---|
| Horizon | 1.17 mm |
| Density | 2.20 g/cm$^3$ |
| Bulk Modulus | 14.90 GPa |
| Shear Modulus | 8.94 GPa |
| Critical Stretch | 0.0005 |

Table 2: Constitutive parameters for the projectile, modeled using the *linear peridynamic solid* constitutive model. No bond failure law was assigned to the projectile.

| Parameter | Value |
|---|---|
| Horizon | 1.17 mm |
| Density | 7.70 g/cm$^3$ |
| Bulk Modulus | 160.00 GPa |
| Shear Modulus | 78.30 GPa |

## 9.2 Quasi-static simulation of a tensile test

A peridynamic simulation of a tensile test experiment is presented in Figure 3. The simulation replicates a standard laboratory test for material characterization [1]. The simulation was carried out using implicit quasi-static time integration (Section 8.3). The bar was modeled using an elastic correspondence constitutive model [34]. As described by Littlewood [18], and Tupek and Radovitzky [36], the formulation for correspondence constitutive models given by Silling, *et al.* [34], may exhibit unphysical low-energy modes of deformation. An additional stabilization term was applied in the tensile test simulation to mitigate the impact of these low-energy modes on the solution [18]. Constitutive model parameters for the




(a) 0.0 µs. (b) 44.0 µs. (c) 131.0 µs.

Figure 2: Explicit dynamic simulation of brittle fracture resulting from impact. The color scale denotes damage, which is defined as the percentage of broken bonds for a given material point.

Table 3: Constitutive parameters for the tensile bar. The projectile was modeled using a linear elastic correspondence constitutive law.

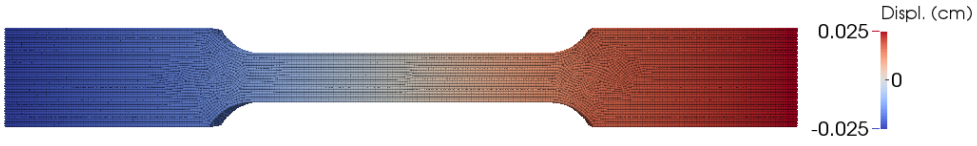
| Parameter | Value |
|---|---|
| Horizon | 0.095 25 cm |
| Density | 8.00 g/cm$^3$ |
| Young's Modulus | 180.00 GPa |
| Poisson's Ratio | 0.30 |
| Stabilization Coefficient | 0.02 |

bar are given in Table 3. The length of the tensile bar is 10.16 cm, with a maximum width of 1.27 cm, a minimum width of 0.635 cm, and a thickness of 0.315 cm. The discretization was created by converting an unstructured hexahedral mesh, created using *Cubit*, to a meshfree representation containing approximately 99 000 nodes. The simulation was divided into four load steps. At each load step, the nonlinear solver was applied to reduce the residual by approximately nine orders of magnitude. The tensile bar simulation was run in parallel using ten 3.0 GHz Intel Xeon processors, requiring a total run time of approximately 1.3 hours.

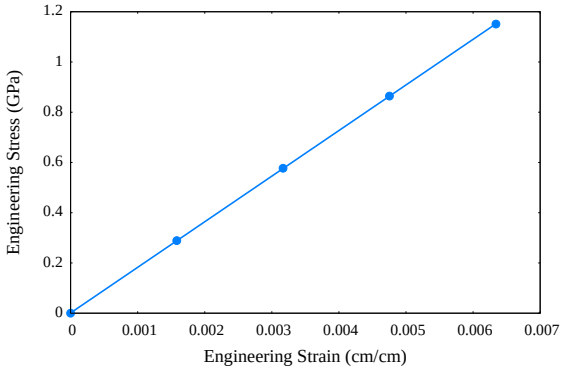
Boundary conditions were applied in the form of prescribed displacements over volumes at the ends of the bar. The volumes over which prescribed displacements were applied were defined as spanning from each end of the bar to a distance equal to two times the horizon inward from the end of the bar. Nodes within these volumes were assigned an initial displacement of zero and a final displacement equal to $0.005\,x$, where $x$ is the distance from the center of the bar (the origin). The result is a prescribed linear displacement field corresponding to an engineering strain of 0.5% applied to the end portions of the bar. This displacement field is an approximation (educated guess that disregards the variable cross section of the bar) of the true displacement field that would occur in a physical tensile bar experiment. It is expected that, due to the approximate nature of the boundary conditions, unphysical artifacts will be present in and around the end segments of the bar. These artifacts are are not expected to significantly pollute the solution in the narrow section of the bar.

The tensile bar simulation was designed to mimic a standard engineering tensile test experiment for material characterization. Young's modulus and Poisson's ratio may be determined experimentally by monitoring the elongation of small segments of the bar using strain gauges and tracking the total force applied to the bar using a load cell. These data can then be used to compute the engineering strain (change in length of the strain gauge divided by its initial length) in both the lateral and longitudinal directions as well as the engineering stress (total force divided by the initial cross-sectional area of the narrow section of the bar). The Young's modulus is computed as the ratio of the engineering stress to the engineering strain in the longitudinal direction. Poisson's ratio is computed as the ratio of the lateral strain to the longitudinal strain. This procedure was modeled within the peridynamic simulation by tracking the displacements of pairs of nodes as well as the net reaction forces in the end segments of the bar. One pair of nodes was chosen to approximate the positions

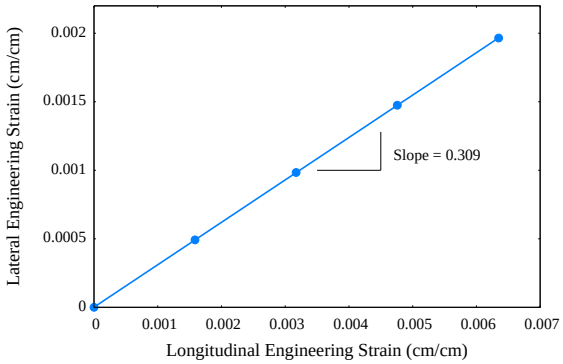
at which the ends of a 2.54 cm strain gauge would be located, assuming the strain gauge were aligned in the longitudinal direction and centered along the length of the bar. A second pair of nodes was chosen to approximate a lateral strain gauge spanning a narrow 0.54 cm section of the bar centered in the longitudinal direction. The resulting engineering stress-strain curve is presented in Figure 3b. The value of Young's modulus calculated in this manner was 181.5 GPa, which differs from the prescribed value of 180.0 GPa by 0.83%. The Poisson's ratio was calculated to be 0.309, which differs from the prescribed value of 0.30 by 3.0%.




(a) Displacement in the loading direction.




(b) Computed engineering stress-strain curve. The recovered elastic modulus is 181.5 GPa.




(c) Computed lateral and longitudinal strains. The recovered Poisson's ratio is 0.309.

Figure 3: Quasi-static simulation of a tensile test.

This page intentionally left blank.

# 10 Summary

The application of peridynamics for computational simulation depends critically on an efficient and robust software implementation. In the preceding sections, a roadmap was presented for implementation of the meshfree approach of Silling and Askari [33], which is the numerical technique used for the vast majority of engineering peridynamic simulations to date. Codes developed at Sandia National Laboratories, including *EMU*, *Peridigm*, *LAMMPS*, and *Sierra/SolidMechanics*, contain peridynamics functionality based on this approach.

The state of computational peridynamics codes reflects the level of maturity of peridynamics as a whole. While great strides have been made since the introduction of peridynamics by Silling in 2000 [31], numerous open research areas remain. With respect to computational simulation, alternatives to the meshfree approach of Silling and Askari should be explored, including alternative discretization techniques, methods for higher-order quadrature, and Eulerian or Arbitrary Lagrangian-Eulerian (ALE) formulations. Constitutive models, contact models, and bond-failure models remain significantly limited, particularly in the context of implicit time integration. In addition to theoretical and numerical considerations, widespread adoption of peridynamics as a tool for engineering analysis requires improvements in the performance and usability of computational peridynamics codes. Practitioners accustomed to mainstream finite-element analysis tools would benefit from pre- and post-processing tools designed specifically for peridynamic models, for example for the specification of volumetric regions for the application of boundary conditions.

Nonlocality pervades many aspects of a peridynamic simulation code. This is apparent within the internal force evaluation, where constitutive model routines must traverse a neighbor list data structure, and in the tangent stiffness matrix, which is significantly more dense that its counterpart in the local theory. The material in the preceding sections draws from experience implementing both standalone peridynamics codes and peridynamics functionality within larger software packages. It is hoped that the material presented will provide clarity on the implementation of peridynamic theory, as presented in the literature to date, in a computational simulation code.

This page intentionally left blank.

# References


[1] ASTM Standard E8 / E8M-13a. Standard test methods for tension testing of metallic materials. ASTM International, West Conshohocken, PA, 2013.

[2] T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear finite elements for continua and structures*. John Wiley & Sons, Ltd., Chichester, England, 2000.

[3] M. Berger and S. Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Transactions on Computers*, C-36(5):570–580, 1987.

[4] F. Bobaru and D. Y. Ha. Adaptive refinement and multiscale modeling in 2D peridynamics. *International Journal for Multiscale Computational Engineering*, 9(6):635–660, 2011.

[5] E. G. Boman, U. V. Catalyurek, C. Chevalier, and K. D. Devine. The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: partitioning, ordering, and coloring. *Scientific Programming*, 20(2), 2012.

[6] M. D. Brothers, J. T. Foster, and H. R. Millwater. A comparison of different methods for calculating tangent-stiffness matrices in a massively parallel computational peridynamics code. *Computer Methods in Applied Mechanics and Engineering*, 279:247–267, 2014.

[7] S.-W. Chi, C.-H. Lee, J.-S. Chen, and P.-C. Cuan. A level set enhanced natural kernel contact algorithm for impact and penetration modeling. *International Journal for Numerical Methods in Engineering*, 2014.

[8] R. Courant, K. O. Friedrichs, and H. Lewy. Über die partiellen differenzensleichungen der mathematischen physik. *Mathematische Annalen*, 100(1):32–74, 1928.

[9] *Cubit* mesh generation code, 2014. `http://cubit.sandia.gov`.

[10] J. T. Foster, S. A. Silling, and W. Chen. An energy based failure criterion for use with peridynamic states. *Journal for Multiscale Computational Engineering*, 9:675–687, 2011.

[11] J. Grandy. Efficient computation of volume of hexahedral cells. Technical Report UCRL-ID-128886, Lawrence Livermore National Laboratory, Livermore, CA, 1997.

[12] A. Griewank and A. Walther. *Evaluating derivatives: Principles and techniques of algorithmic differentiation, second edition*. Frontiers in Applied Mathematics. SIAM, Philadelphia, PA, 2008.

[13] S. F. Henke and S. Shanbhag. Mesh sensitivity in peridynamic simulations. *Computer Physics Communications*, 185:181–193, 2014.

[14] T. J. R. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Englewood Cliffs, N.J. : Prentice-Hall, 1987.

[15] A. Katiyar, J. T. Foster, H. Ouchi, and M. M. Sharma. A peridynamic formulation of pressure driven convective fluid transport in porous media. *Journal of Computational Physics*, 261:209–229, 2014.

[16] T. A. Laursen. *Computational contact and impact mechanics: fundamentals of modeling interfacial phenomena in nonlinear finite element analysis*. Springer-Verlag, Heidelberg, Germany, 2002.

[17] D. J. Littlewood. Simulation of dynamic fracture using peridynamics, finite element modeling, and contact. In *Proceedings of the ASME 2010 International Mechanical Engineering Congress and Exposition (IMECE)*, Vancouver, British Columbia, Canada, 2010.

[18] D. J. Littlewood. A nonlocal approach to modeling crack nucleation in AA 7075-T651. In *Proceedings of the ASME 2011 International Mechanical Engineering Congress and Exposition (IMECE)*, Denver, Colorado, 2011.

[19] D. J. Littlewood, J. D. Thomas, and T. R. Shelton. Estimation of the critical time step for peridynamic models. Presented at the SIAM Conference on Mathematical Aspects of Materials Science, Philadelphia, Pennsylvania, 2013.

[20] R. W. Macek and S. A. Silling. Peridynamics via finite element analysis. *Finite Elements in Analysis and Design*, 43:1169–1178, 2007.

[21] J. A. Mitchell. A nonlocal, ordinary, state-based plasticity model for peridynamics. SAND Report 2011-3166, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2011.

[22] J. A. Mitchell. A nonlocal, ordinary-state-based viscoelasticity model for peridynamics. SAND Report 2011-8064, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2011.

[23] M. L. Parks, R. B. Lehoucq, S. J. Plimpton, and S. A. Silling. Implementing peridynamics within a molecular dynamics code. *Computer Physics Communications*, 179(11):777–783, 2008.

[24] M. L. Parks, D. J. Littlewood, J. A. Mitchell, and S. A. Silling. Peridigm Users' Guide v1.0.0. SAND Report 2012-7800, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2012.

[25] *Peridigm* peridynamics code, 2014. `http://peridigm.sandia.gov`.

[26] E. Phipps and R. Pawlowski. Efficient expression templates for operator overloading-based automatic differentiation. In S. Forth, P. Hovland, E. Phipps, J. Uke, and A. Walther, editors, *Recent Advances in Algorithmic Differentiation*, volume 87 of *Lecture Notes in Computational Science and Engineering*. Springer, 2012.

[27] *Sacado* software package, 2015. `http://trilinos.org/packages/sacado/`.

[28] P. Seleson and D. J. Littlewood. Convergence studies in meshfree peridynamic simulations. *Computers and Mathematics with Applications*. To appear.

[29] P. D. Seleson. Improved one-point quadrature algorithms for two-dimensional peridynamic models based on analytical calculations. *Computer Methods in Applied Mechanics and Engineering*, 282:184–217, 2014.

[30] SIERRA Solid Mechanics Team. Sierra/SolidMechanics 4.36 user's guide. SAND Report 2015-2199, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2015.

[31] S. A. Silling. Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids*, 48:175–209, 2000.

[32] S. A. Silling. Linearized theory of peridynamic states. *Journal of Elasticity*, 99:85–111, 2010.

[33] S. A. Silling and E. Askari. A meshfree method based on the peridynamic model of solid mechanics. *Computers and Structures*, 83:1526–1535, 2005.

[34] S. A. Silling, M. Epton, O. Weckner, J. Xu, and E. Askari. Peridynamic states and constitutive modeling. *Journal of Elasticity*, 88:151–184, 2007.

[35] S. A. Silling, D. J. Littlewood, and P. D. Seleson. Variable horizon in a peridynamic medium. *Journal of Mechanics of Materials and Structures*. To appear.

[36] M. R. Tupek and R. Radovitzky. An extended constitutive correspondence formulation of peridynamics based on nonlinear bond-strain measures. *Journal of the Mechanics and Physics of Solids*, 65:82–92, 2014.

This page intentionally left blank.

DISTRIBUTION:

1 MS 0899 Technical Library, 9536 (electronic copy)

This page intentionally left blank.

Sandia National Laboratories